UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/542,714	04/04/2000	ALLAN HAVEMOSE	AMI 99 0006	5268

32718 7590 12/04/2003

GATEWAY, INC.
ATTN: SCOTT CHARLES RICHARDSON
610 GATEWAY DR., Y-04
N. SIOUX CITY, SD 57049

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2124	12

DATE MAILED: 12/04/2003

Please find below and/or attached an Office communication concerning this application or proceeding.

PRE

Office Action Summary	Application N .	Applicant(s)	
	09/542,714	HAVEMOSE, ALLAN	
	Examiner	Art Unit	
	Tuan A Vu	2124	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
- Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09/11/03*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-44* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-44* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *04 April 2000* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. §§ 119 and 120

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

13)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. § 119(e) (to a provisional application) since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

a) ☐ The translation of the foreign language provisional application has been received.

14)☐ Acknowledgment is made of a claim for domestic priority under 35 U.S.C. §§ 120 and/or 121 since a specific reference was included in the first sentence of the specification or in an Application Data Sheet. 37 CFR 1.78.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449) Paper No(s) _____ .
4) ☐ Interview Summary (PTO-413) Paper No(s). _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: .

DETAILED ACTION

1. This action is responsive to the Applicant's Appeal Brief filed September 11, 2003. As indicated in Applicant's amendments previously filed 3/20/03, claims 1,7,13,18 have been amended, and claims 21-44 added. Claims 1-44 are pending in the office action.

In view of the arguments by Applicants in the Appeal Brief filed 9/11/03, the finality of the previous rejection is herein withdrawn; and a new ground of rejection is herein effected.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-5, 7-11, 13-18, 20-23, 25-28, 30-35, 37-39, 41-42, and 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beadle et al., USPN: 6,530,075 (hereinafter Beadle_1), and further in view of Beadle et al., USPN: 6,295,641 (art of record - hereinafter Beadle_2).

As per claim 1, Beadle_1 discloses a method for dynamic compiling, such method comprising:

loading byte code (e.g. col. 2, lines 38-45) on a data processing processor, i.e. digital information appliance as claimed;

such byte code suitable for instrumentation such as having a tagged section (e.g. Fig. 3; Fig. 2A-B, 5A-B – Note: keywords used to mark what to compile is equivalent to tag);

identifying the tagged section of byte code (e.g. Fig. 2A-B; 5A-B; *keyword* - col. 5, lines 1-25) and compiling such tagged section (e.g. col. 6, line 56 to col. 7, line 27);

wherein the tagged section is compiled when loaded (e.g. *ClassLoader* – Fig. 4; col. 6, line 56 to col. 7, line 27) so as to enable the processor to utilize it without additional compiling (e.g. *passed ... to the interpreter* - col. 9, lines 11-39 - Note: passing methods to the interpreter is equivalent to not re-compiling it).

Beadle_1 further discloses including of a dynamic interactive API (e.g. API – Fig. 5A-5C; col. 6, line to col. 8, line 21; col. 9, lines 40-51) in the executing code wherein the byte code compilation and optimization takes place at JIT time.

But Beadle_1 does not explicitly specify that the byte code includes one dynamic base object, such object comprising an interface dynamic object and an implementation dynamic object communicating with each other over a message bus. However, Beadle_1 discloses a communication system having a bus interconnecting appliances, client or host system memory to the peripheral and network interfaces (e.g. Fig. 1; col. 3, lines 23-45); a runtime environment using object-oriented class loader (e.g. Fig. 4); and a development environment distribution of code to heterogeneous platforms in a network; and desire to spare storage resources therein (e.g. col. 2-col. 3). In view to the teachings by Beadle_1 as to generate a dynamic interactive object API from above, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the client/server application communication paradigm and byte code load/execute runtime environment as presented by Beadle_1 so that a interactive dynamic object (or interface dynamic object as claimed) as taught by Beadle_1 is included in the byte code because this additional dynamic code object would provide interaction with the user for

effecting the resources optimizing of the execution environment, by effecting the translation and execution of code as specified by dynamic user's requests to effect such compilation without errors (Beadle_1: col. 2, lines 14-28).

Nor does Beadle_1 disclose including a dynamic implementation object in the byte code. Analogous to the paradigm of communications suggested by both Beadle_1, Beadle_2, in a method to select byte codes for just in time compilation, discloses a interface object and dynamically create classes and methods in order to accomplish the selective compilation, or dynamic implementation object code (e.g. col. 4, line 59 to col. 45; Fig. 3-5). It would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the client/server application communication paradigm and byte code load/execute runtime environment as presented by Beadle_1 so that the application code thus downloaded to the appliance would further an dynamic implementation base object as suggested by Beadle_2. One skilled in the art would be motivated to do so because this would complement to the dynamic interface object by Beadle_1 in that it dynamically implements and selects classes and methods suitable to be part of the implementation of the appliance code without extraneous storing of ready-made code which would otherwise strain the appliance's storage capacity; and to avoid errors when application used by the device are not selected correctly for execution (Beadle_2: col. 2, lines 4-19).

As for the message bus limitation, Beadle_1's use of platform-independent byte codes (col. 4, lines 30-42), or Beadle_2's intent for distributed applets (e.g. col. 2, lines 10-20) already suggests the suggests the application code such as to be distributed over the network. In view of the above suggestion to effect distributions and downloads through the Web, the limitation of

including a message bus would have been obvious for one of ordinary skill in the art at the time the invention was made would be motivated to include message bus for passing electronic data between network-linked machines as suggested above because communications in network necessarily include passing of message content between sending and receiving appliances.

As per claim 2, Beadle_1 further discloses encoding application source code in byte-code, such byte-code suitable for further analysis (e.g. col. 4, lines 25-52); and tagging a section (re claim 1).

As per claim 3, Beadle_1 further discloses that the byte-code is suitable for instrumentation (e.g. *prediction* - col. 10, lines 4-15; Fig. 8); suitable for further analysis including compiler optimization (e.g. *hot spots* - Fig. 3), for interpreters (e.g. *interpreter* - col. 4, lines 40-42, 50-52), and for use in generating binary code (e.g. col. 6, lines 50-52) for the digital information appliance.

As per claim 4, Beadle_1 further discloses tagging byte-code for performance sensitive instrumentation (e.g. *hot spots* - Fig. 3).

As per claim 5, Beadle_1 further discloses persistent storage of tagged sections of byte-code (e.g. Fig. 1; col. 3, lines 58-62).

As per claim 7, Beadle_1 discloses a digital information appliance for dynamic compiling, comprising

a processor for implementing a program;

a memory for storing program instructions (e.g. Fig. 1; col. 3, lines 58-62), said program having instructions to load byte-code (e.g. *classLoader* – Fig. 4), such byte-code being suitable

for having a tagged section (e.g. Fig. 2A-B, 5A-B – Note: keywords used to mark what to compile is equivalent to tag);

identifying the tagged section (e.g Fig. 2A-B, 5A-B; *keyword* - col. 5, lines 1-25) and compiling the tagged section (e.g. *JITEnabled* - Fig. 2A-B, 5A-B); wherein

the tagged section is compiled when loaded (e.g. *ClassLoader* – Fig. 4; col. 6, line 56 to col. 7, line 27) so as to enable the processor to utilize it without additional compiling (e.g. *passed ... to the interpreter* - col. 9, lines 11-39 - Note: passing methods to the interpreter is equivalent to not re-compiling it).

But Beadle_1 fails to expressly specify that the byte code includes one dynamic base object, such object comprising an interface dynamic object and an implementation dynamic object communicating with each other over a message bus. However, this limitation has been addressed in claim 1 above, hence is rejected herein using the same rationale as set forth therein.

As per claims 8-11, these are the apparatus version of claims 2-5 above, respectively, hence incorporates the corresponding rejections of those claims respectively.

As per claim 13, Beadle_1 discloses a system for an execution environment suitable for dynamic compilation, comprising

a memory device (e.g. Fig. 1; col. 3, lines 58-62);

a loader coupled with the memory device for loading byte-code (e.g. *classLoader* – Fig. 4), such byte-code suitable for suitable for having a tagged section (e.g. Fig. 2A-B, 5A-B – Note: keywords used to mark what to compile is equivalent to tag); the loader being capable of interpreting (e.g. *interpreter* - col. 4, lines 40-42, 50-52), just in time compiling (e.g. Fig. 2A-B, 5A-B), and pre-compiling (Note: byte code submitted to the JIT is equivalent to having an

inherent pre-compilation performed in order to have such byte-code ready for the JIT 410 of Fig. 4)

an identifier coupled with the loader for identifying the tagged section (*keyword* - col. 5, lines 1-25);

a compiler (e.g. Fig. 4) coupled with the identifier,

wherein the tagged section is compiled when loaded so to enable the tagged section to be utilized without additional compiling (e.g. col. 6, line 56 to col. 7, line 27; *passed ... to the interpreter* - col. 9, lines 11-39).

As per claim 14, Beadle_1 further discloses

an encoding means to encode application source code to byte-code in an processor-independent form suitable for further analysis (e.g. Fig. 2-5; col. 4, lines 25-52; *performance results* - col. 6, lines 11-18 – Note: byte code implicitly discloses the existence of encoding means);

a tagger for tagging a section of the byte-code (e.g. Fig. 2A-B, 5A-B – Note: keywords used to mark what to compile is equivalent to tag).

As per claims 15-16, these are the system claims corresponding to claims 3 and 4, respectively, hence incorporate herein all the corresponding rejections therein.

As per claim 18, Beadle_1 discloses a method for providing an execution environment in an information appliance network (Fig. 1), comprising

encoding an application source code in a processor-independent byte-code (e.g. Fig. 2-5; col. 4, lines 25-52; *performance results* - col. 6, lines 11-18 – Note: byte code implicitly discloses the existence of encoding means);

tagging (e.g. Fig. 2A-B, 5A-B – Note: keywords used to mark what to compile is equivalent to tag) at least some portion of such byte-code; and

compiling some portion of such tagged byte-code (e.g. *ClassLoader* – Fig. 4; col. 6, line 56 to col. 7, line 27).

But Beadle_1 fails to specify that the byte code includes one dynamic base object, such object comprising an interface dynamic object and an implementation dynamic object communicating with each other over a message bus. However, this limitation has been addressed in claim 1 above, hence is rejected herein using the same rationale as set forth therein.

As per claim 20, Beadle_1 further discloses identifying of the tagged portion of byte-code (e.g. Fig. 2A-B, 5A-B – Note: keywords used to mark what to compile is equivalent to tag).

As per claim 21, Beadle_1's use of platform-independent byte codes (col. 4, lines 30-42), or Beadle_2's intent for distributed applets (e.g. col. 2, lines 10-20) already suggests the application code such as to be distributed over the network. In view of the above suggestion to effect distributions and downloads through the Web, the limitation of including a message bus would have been obvious as set forth in claim 1 because communications in network inherently include passing of message content between sending and receiving appliances.

As per claim 22, this inter-processor message bus limitation (re claim 21) for communication over the network (re claim 22) would have been obvious with reference to claim 1 above for the same rationale set forth therein.

As per claim 23, see Beadle_1: col. 1, line 40 to col. 2, line 14.

As per claim 25, referring to claim 1, only Beadle_2 teaches dynamic implementation object and interface object (e.g. Fig. 3-5 – Note: interface object for user specification event,

implementation object for creating class instances on the fly). By virtue of the teachings of Beadle_1 for using a interactive API dynamic extension class object used in the rejection of claim 1, the bi-directional between interface object and implementation object limitation herein would have been obvious using the same rationale as mentioned therein, and also because the interface object communicating with the implementation object is inherent to how the final executable can be generated and delivered.

As per claims 26-28, these are the apparatus claims corresponding to claims 21-23 from above, respectively, hence incorporate herein all the corresponding rejections therein.

As per claim 30, this bi-directional limitation has been addressed in claim 25 above; hence is rejected herein using the same rationale set forth therein.

As per claim 31, this dynamic base object limitation has been addressed in claim 1 above; hence is rejected herein using the same rationale set forth therein.

As per claims 32-35, these are the system claims corresponding to claims 21, 22, 25, and 23, respectively, hence incorporate herein all the corresponding rejections therein.

As per claims 37-39, these are method claims corresponding to claims 21-23 from above, respectively, hence incorporate herein all the corresponding rejections therein.

As per claim 41, this method claim incorporates the rejection of claim 25 above.

As per claim 42, Beadle_1 discloses multiple lines of code for the dynamic base object (e.g. col. 5) and Beadle_2 discloses interface object/class for effecting the interface object interaction (e.g. Fig. 2-5).

As per claim 44, Beadle_1 discloses a safe multithreads Java-based environment (e.g. Fig. 2-5). Using Java code with its inherent object instantiation and safe multi-threaded object execution implicitly discloses the safe nature of such multi-threaded environment by Beadle_1.

4. Claims 24, 29, 36, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beadle_1, USPN: 6,530,075 and Beadle_2, USPN: 6,295, 641, as applied to claims 1, 7, 31, and 18; and further in view of and Crelier, USPN: 6,151,703 (hereinafter Crelier).

As per claim 24, Beadle_1 and Beadle_2 does not discloses that the dynamic base object is programmed through a scripting language but suggests network communication for data processing (e.g. Beadle_1: col. 3, lines 20-46) and a distribution of applets (e.g. Beadle_2: col. 2, lines 10-20). Crelier, in a method to effect distribution of code using optimized compilation with JIT similar to the optimized code loading by Beadle_1, discloses a dynamic implementation base object similar to that of Beadle_1 or Beadle_2 for dynamically instantiating methods and class (Crelier: col. 7, line 61 to col. 11, line 40) and further suggests use of scripts (e.g. col. 3, lines 43-47; *script 205* - col. 6, lines 5-18). In view of the combined suggestion by Beadle_1/Beadle_2 and teachings by Crelier, it would have been obvious for one of ordinary skill in the art at the time the invention was made to use the Java-written application suggested by Beadle_1 (combined with Beadle_2) and additionally implement it with the browser technologies such as scripting language to implement object to implement a Java-based application as taught by Crelier because of the common practice known in Web-based client/server transaction and also of the speedy script interpretation, thus improve time and resource efficiency, e.g. less compiling time, and take full advantage of Browser given technologies and secure protocol, e.g. CGI gateway security.

As per claim 29, this is the apparatus claim corresponding to claim 24 from above, respectively, hence incorporates herein all the corresponding rejection therein.

As per claim 36, this claim incorporates the rejection of claim 24 above.

As per claim 40, this method claim incorporates the rejection of claim 24 above.

5. Claim 43 is rejected under 35 U.S.C. 103(a) as being unpatentable over Beadle_1, USPN: 6,530,075 and Beadle_2, USPN: 6,295, 641, as applied to claim 1, and further in view of Wu et al., USPN: 5,987,256 (art of record - hereinafter Wu).

As per claim 43, Beadle_1 suggests optimizing code for distributed process code (e.g. col. 2, line 40 to col. 3, line 19) and Beadle_2 also discloses distribution of applets (col. 2, lines 10-20). Further, Wu in a method to provide an interface or interactive application object using JIT and optimizing HTTP code similar to that described by Beadle_2, discloses a thin appliance (e.g. col. 1, lines 17-52). It would have been obvious for one of ordinary skill in the art at the time the invention was made to include in Beadle_1's (in combination w/ Beadle_2) types of appliances a thin appliance as suggested by Wu, because such thin appliance is even more restricted in resources and storage capacity as Beadle_1's suggested optimizing of resources, thus would require the same optimization schemes as disclosed by Beadle_1, or Beadle_2.

6. Claims 6, 12, 17, and 19 are rejected under 35 U.S.C. 103(a) as being unpatentable over Beadle_1, USPN: 6,530,075 and Beadle_2, USPN: 6,295, 641, as applied to claims 1, 7, 31, and 18, and further in view of Beadle et al, USPN: 6,305,012 (hereinafter Beadle_3).

As per claim 6, Beadle_1 in combination with Beadle_2 does not expressly specify validating whether the byte-code conforms with the byte-code suitable for the application of the digital information processor/appliance. Beadle_1 discloses checking byte-code for threshold

and a safe class loading and compiling (e.g. Fig. 6-8); and Beadle_2 checks compilation commands to prevent executing error-prone sections using a data structure comparison (e.g. Fig. 6). Further, Beadle_3, in a method to provide tag-based selective compilation of code analogous to the method of tag/skip JIT compilation of Beadle_1 in the combination of claim 1, discloses validating that the code as sent to the JIT conforms with the operating system for the application in the digital information appliance (e.g. Fig. 7; *operating system* - col. 8, lines 12-40). As suggested by Beadle_2, the secure operation of downloaded or received data ought to be validated prior to or during usage of such data, and in view of such teachings, it would have been obvious for one of ordinary skill in the art at the time the invention was made to ensure that the downloaded byte code as taught by Beadle_2 (combined with Beadle_1 as in claim 1) is checked for compatibility or persistency with the operating system of the appliance (i.e. being suitable as claimed) as taught by Beadle_3, in order to avoid system crashes or faults during executing/using of such code, which would considerably undermine the intent of optimizing and alleviating resources in the target appliance as originally intended by Beadle_1.

As per claim 12, this claim the apparatus version of and corresponds to claim 6 above, hence incorporates the corresponding rejection set forth therein.

As per claim 17, this claim is the system version of and corresponds to claim 6 above, hence incorporates the corresponding rejection set forth therein.

As per claim 19, this claim includes a limitation similar the limitation of claim 6 above, hence incorporates the corresponding rejection set forth therein.

Examiner's remarks

7. In reference to the interview with agent Malinowski # 43,423, effected 11/21/03, Examiner has the following remarks:

First, Examiner has sought advice from Applicants as to seek to modify the claims so that the limitation 'wherein the byte code includes at least one dynamic base ... over a message bus" (re claim 1, 7, 18) would relate to the upper part of all independent claims in such a way as to make the optimized compiling and linking process as claimed clearly distinguishable over the prior art.

Second, after consideration of the proposed modifications by Applicants over the phone, Examiner deems that such modifications are not sufficient to render the independent claims such to overcome the art of record.

Third, regarding the above limitation, it is noted that although such 'dynamic base object' further specifies the byte code as claimed, it does not explain how it enhances or renders the process of compiling and optimizing so that it particularly distinguishes over prior known techniques, for it appears as though the 'dynamic base object' describes an additional subject matter irrelevant to the compiling process; if not saying that it barely specifies an intended use and bears no patentable weight over the main invention as claimed.

However, in regard to the Appeal Brief filed 9/11/03 and after consideration of the pertinent arguments presented by Applicants, the finality of the last office action has been withdrawn.

Again, Examiner likes to repeat that only when the claims are amended so that the 'dynamic base object', as presented above, is meaningfully linked to and correlates with the

optimization steps of the compiling process which is the core of the invention, then the relevant claims **may** (emphasis added) be in better position for allowance.

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (703)305-7207. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (703)305-9662.

Any response to this action should be mailed to:

Commissioner of Patents and Trademarks

Washington, D.C. 20231

or faxed to:

(703) 872-9306 (for formal communications intended for entry)

or: (703) 746-8734 (for informal or draft communications, please label "PROPOSED" or "DRAFT")

Hand-delivered responses should be brought to Crystal Park II, 2121 Crystal Drive, Arlington. VA. , 22202. 4th Floor(Receptionist).

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the receptionist whose telephone number is (703) 305-3900.

Kakali Chaki

VAT
November 25, 2003

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100